BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI
CS/SS G527 Cloud Computing
I Semester 2016-17
Assignment-2
Weightage: 20% (100M) **Due Date of Submission:** 24-Nov-2016

==========

Important to Note:

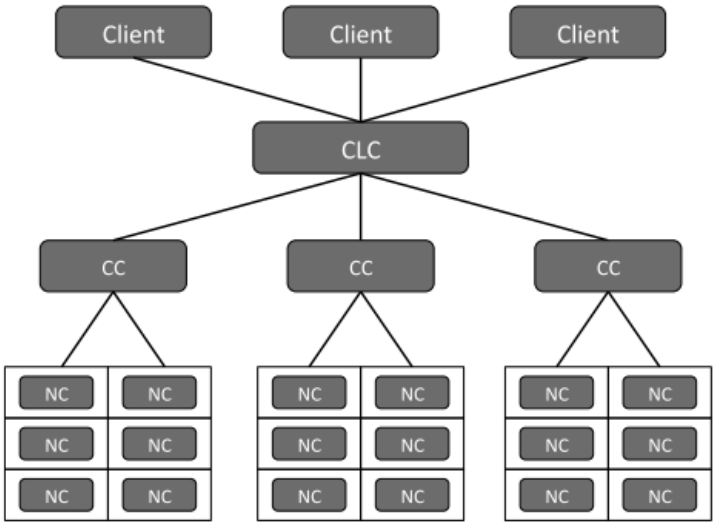
1. Group of maximum 4 students.
2. 3 programming problems.
3. For any clarifications please contact me (khari@pilani.bits-pilani.ac.in).

Plagiarism will be thoroughly penalized.

==========

P1. [Implementing simplified IaaS model]

Your are required a build a cloud with at least two clusters. Each cluster has two physical machines (PM). Every PM runs a Node Controller (NC). Each cluster is considered as one availability zone. Each cluster has a Cluster Controller (CC). There is one Cloud Controller (CLC) which interfaces with users, and admin. Schematic diagram of cloud is shown below. The structure is based on eucalyptus cloud system. You can see its source code but can't use it in your assignment.



a) Implement NC, CC and CLC for the following roles.
NC: runs on each PM. responsible for creating, stopping, resuming or deleting VM on PM. When CC requests, NC sends available resources (cores, memory and disk). It uses *libvirt* to interact with KVM.
CC: CC runs on one of the PM. When CLC sends a request (with number of VM instances and *type* (cores, disk and memory)), CC requests all NCs to send available resources. If resources match with *type* requested, it will use a scheduling algorithm to select PCs where they can be instantiated. If resources do not match, it will send failure status to CLC.
CLC: Offers REST API/CLI to users to create VMs and his delete VMs. When VM is created, user specifies number of cores, RAM, disk space required and availability zone. He can also set auto-scale option in terms of increasing(or decreasing) one VM for every 20% increase (decrease) over 500 requests per minute.

b) Measure the performance of cloud resource utilization for each of the scheduling algorithms CC uses. (i) greedy algorithm (ii) round robin algorithm (iii) match

making algorithm. Performance has to be plotted in these graphs: (i) number of VMs per PM vs time (ii) CPU utilization per PM vs time (iii) memory used per PM vs time

Assumptions: There is only one user. ISO image of Ubuntu/anyother (to be loaded into VM) is configured with Apache web server, and runs after booting with default page. User enables Auto scale option. Traffic is generated using httpmon tool.

c) Assume that CC uses round-robin algorithm for scheduling. CLC runs a procedure every 2 minutes to consolidate servers. CLC uses first-fit bin-packing algorithm to consolidate servers (VMs). It uses VM migration facility of KVM (through CC & NC) in case VM has to be shifted to another PM. Measure the results. Performance has to be plotted in these graphs: (i) number of VMs per PM vs time (ii) CPU utilization per PM vs time (iii) memory used per PM vs time

Deliverables:

- Brief Design Documents for (a) to (c). (typewritten) in design.doc/.pdf
- Code in folder named Code
- Test cases in testcases.txt
- Performance Plots in plots.doc/.pdf separately for (b) and (c)

[25+15+10=50M]

P2. [Implementation of simplified SaaS model on top of P1 (a)]

a) Assuming that the hard disk attached to a VM by default, is transient. Now offer service similar to EBS. User should be able to attach disks of chosen capacity to VM. When VM stops these disks are still valid and can be attached to any VM. Every disk created will have a replica in another PC in the same availability zone. Any changes done to disk, will be reflected there.

b) Here implement a object storage service. Reserve disk space in each PM. Let this be called as a bucket. When user specifies a name and object, that name is hashed and stored in a bucket corresponding to hash (as a file). Similar way user retrieves the file. Every object should have a copy stored in a bucket on another availability zone. Offer REST API like GET, PUT and DELETE to deal with the objects.

[10+10=20M]

Deliverables:

- Brief Design Documents for (a) and (b) (typewritten) in design.doc/.pdf
- Code in folder named Code
- Test cases in testcases.txt

P3. Choose a container library, say LXC. Implement checkpointing and recovery feature at the library level. Develop CLI to checkpoint a process running in a container and restore it on a different system. Checkpointing should include process address space, signal mask , file descriptors (only regular files) and CPU state. Recovery command should restore entire process.

Deliverables:

- Brief Design Documents (2 pages, typewritten) in design.doc/.pdf
- Code in folder named Code
- Test cases in testcases.txt

[30M]

How to upload?

- Create group.txt file and put idno, name of members into this file.
- Make a directory for each problem like P1, P2 etc and copy deliverables into these directories.
- Tar all of them including group.txt into idno1_idno2_idno3_assignment2.tar
- Upload at the course webpage.

===End of assignment===